Trends in Parallel Computing

Oliver A. McBryan

CU-CS-507-90 December 1990

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **1990**	2. REPORT TYPE	3. DATES COVERED **00-00-1990 to 00-00-1990**
4. TITLE AND SUBTITLE **Trends in Parallel Computing**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Colorado at Boulder,Department of Computer Science,Boulder,Co,80309**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**17**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Trends in Parallel Computing

Oliver A. McBryan

CU-CS-507-90 December 1990

Department of Computer Science
University of Colorado at Boulder
Campus Box 430
Boulder, Colorado 80309-0430 USA

(303) 492-7514
(303) 492-2844 Fax
mcbryan@boulder.colorado.edu

TRENDS IN PARALLEL COMPUTING

Oliver A. McBryan†
Center for Applied Parallel Processing (CAPP)
Department of Computer Science
University of Colorado
Boulder, CO 80309, USA.

Abstract:

A general consensus is developing that highly parallel computers are the only practical near term route to the teraflops of computing power required by many scientific and engineering problems. While current conventional supercomputers already utilize low parallelism, highly parallel computers are still in the experimental stage, although they are increasingly visible at the front line of grand challenge computations. While it has been clearly demonstrated that such systems can be built, and are reliable, there has been only limited success with developing appropriate user-level software.

We survey recent and upcoming developments in the area of parallel supercomputing, focusing on the the underlying trends that have proved successful. We illustrate the general remarks with more detailed discussion of a few representative systems. Finally we discuss the likely next-generation computing environment: a heterogeneous ensemble of scalar, vector, parallel and graphics computers along with disk farms and network interfaces to users.

† Supported in part by Air Force Office of Scientific Research, under grant AFOSR-89-0422.

1. OVERVIEW OF PARALLEL SYSTEMS

1.1. Classification of Parallel Computers

Parallel computers may be broadly categorized in two types - SIMD or MIMD[1]. SIMD and MIMD are acronyms for *Single Instruction stream - Multiple Data stream,* and *Multiple Instruction stream - Multiple Data stream* respectively. In SIMD computers, every processor executes the same instruction at every cycle, whereas in a MIMD machine, each processor executes instructions independently of the others. The vector unit of a CRAY computer is an example of SIMD parallelism - the same operation must be performed on all components of a vector. Most of the interesting new parallel computers are of MIMD type which greatly increases the range of computations in which parallelism may be effectively exploited using these machines. However, this occurs at the expense of programming ease - MIMD computers are much more difficult to program than SIMD machines. Many current designs incorporate both MIMD and SIMD aspects - typically each node of an MIMD system is itself a vector processor.

Another easy categorization is between machines with global or local memories. In local memory machines, communication between processors is entirely handled by a communication network, whereas in global memory machines a single high-speed memory is accessible to all processors. Beyond this, it becomes difficult to categorize parallel machines. There is an enormous variety in the current designs, particularly in the inter-connection networks. For a taxonomy of designs, see the paper of Schwartz[2].

While many interesting parallel machines involve only a few processors, we will concentrate in this survey on those machines which have moderate to large numbers of processors. Important classes of machines such as the CRAY Y-MP, CRAY-2 and the Japanese supercomputers are therefore omitted from the subsequent discussions.

1.2. A Partial List of Multi-processors

There are at least 100 recent or current parallel computer projects worldwide. Table 1 lists a selection of such projects, a mix of university and industrial enterprises. This is just a sample of the diverse projects but covers a wide range of different architectures chosen more or less at random. While some of these projects are unlikely to lead to practical machines, a substantial number will probably lead to useful prototypes. Several commercial parallel computers have been or are already in production (e.g., Connection Machine CM-2, Denelcor HEP-1, Evans and Sutherland ES-1, FPS T-Series, ICL DAP, Intel iPSC2, Meiko, Myrias SPS-2, NCUBE-2, Parsytec, SUPRENUM-1, Symult 2010) and more are under development. Some of these products have been commercial failures (e.g. Denelcor HEP, ETA-10, ES-1, FPS T-Series, Symult 2010), yet they have provided important insights into parallelism. One should also remember that the latest CRAY computers, (e.g. CRAY Y-MP and CRAY-2) involve multiple processors, and other vector computer manufacturers such as NEC, Fujitsu and Hitachi have similar strategies.

Beyond the simple classification into SIMD or MIMD computers we recognize a vast array of different approaches to the task of building a parallel architecture. We will now look at the

Table 1: Some Parallel Computer Projects

ICL DAP	CalTech Hyper-Cube
Intel iPSC hypercube	NCUBE hypercube
Denelcor HEP-1	NYU/IBM Ultra-computer/RP3
Connection Machine CM-2	FPS T-Series
CRAY Y-MP and CRAY-2	ETA-10
IBM 3096	Multiflow
Goodyear MPP	MIT Data-flow Machines
BBN Butterfly	Wisconsin Database Machine
SUPRENUM-1	IBM GF-11 and TF1
Paralex Pegasus	Symult 2010
Myrias SPS-2	Cedar Project
Meiko	Parsytec
Evans & Sutherland ES-1	CMU iWarp
Flex	Alliant FX-8
Sequent Balance	Encore Multimax
CCI Navier-Stokes Machine	TERA
MasPar MP-1	Kendall Square

reasons for this broad range by discussing some of the possibilities encountered for both node and communication facilities.

1.3. Node Design

By *node* we mean the individual computational processor, along with its associated communications hardware, and local memory if available. Node design tends to be far less variable than other aspects of parallel computers. The main reason for this is that most architects have relied on off-the-shelf products for the node - standard microprocessors, floating point accelerators and memory chips. The advantage is that startup time for a project may be substantially reduced. Additionally there is a usually a substantial body of low-level software available for such processors - software such as compilers, assemblers and debuggers. Thus we find that an enormous number of the current parallel computer products are based on one or more of the Intel 80386, Motorola 68020, INMOS T800 transputer, the Weitek floating point accelerators, and (recently) the Intel i860. Typically one of these microprocessors will be combined on a board with a floating point coprocessor (e.g. 80387 or 68881), possibly a Weitek processor and several megabytes of memory. Memory consumes substantial space, and current systems have in the range of 1 to 32 Mbytes per node. Despite these general comments, it should be mentioned that some manufacturers have developed custom processors specifically for parallel computers. In the list above we would point to the DAP, NCUBE, HEP-1, CM-2, ES-1, iWarp and Navier-

Stokes machines as examples.

1.4. Communication Features

The range of inter-processor communication facilities is what really characterizes the differences in architecture among the various parallel machines. While we have previously distinguished the shared memory and distributed memory classes, one should observe that this distinction should not be taken too seriously. A distributed memory computer can certainly simulate a global shared memory and vice versa.

Communication pathways are typically built either from direct point-to-point connections, or from busses. Busses have the advantage that many processors may be serviced by one communication path, but have the disadvantage of slower bandwidth performance as the number of processors increases. With point-to-point connections, processors that are directly connected will have very efficient communication, but indirectly connected processors will likely incur substantial extra overheads including increased latency as well as lower bandwidth.

The most popular interconnection strategies involve simple symmetric arrangements including rings, meshes, hypercubes, trees and complete connections or crossbars. The prevalence of hypercube designs is explained by the fact that the architecture supplies substantial parallel bandwidth for many standard algorithms, for example the Fast Fourier Transform, while at the same time incurring only relatively modest fan-in and fan-out of connections which grow in number only logarithmically with the processors. On the other hand, hypercube wiring complexity grows with the number of processors and the likelihood increases that most wires are unused most of the time. Table 2 compares several simple topologies as a function of processor number P from the point of view of amount of *wiring* (difficulty of building), *connectivity* (ease of programming) and *maximal path* (efficiency of long-range communication).

Table 2: Properties of Interconnection Networks

Network	Wires	Connectivity	**Max Path**
Cross Bar	P^2	P	1
1D Grid	P	2	P
2D Grid	$2P$	4	$2\sqrt{P}$
Binary Tree	P	1–3	$2\log P$
Hypercube	$.5P\log P$	$\log P$	$\log P$

While cross bar switches are extremely difficult to build for large numbers of processors, they have tremendous flexibility in terms of efficiency and ease of use. It is conceivable that a technological breakthrough such as optical switching might allow cross bars to be built that would connect thousands of processors. For the time being, crossbars are restricted to small systems of at most 64 processors, or to providing interconnects among the processors of subclusters within larger machines (e.g. the ES-1).

Bus based connection networks are attractive for moderate numbers of processors, for example 16 to 32. Beyond this point bandwidth begins to suffer intolerably. Architectures based on busses therefore tend to be hierarchical beyond that number of processors. As an example, the SUPRENUM-1 computer uses a fast local bus to connect within a cluster of 16 processors. Clusters are arranged in a rectangular grid and connected by row and column busses, which has the added attraction of providing redundancy and double bandwidth. The Myrias SPS-2 is similar, utilizing three levels of bus: 4 processors connected on a single board by a bus, cages of 16 boards connected by a pair of backplane busses, and finally cages connected by third-level busses.

New configurations of processors continue to be proposed. Of particular interest are Giloi and Montenegro's TICNET architecture[3], and Faber's vertex-symmetric minimal path networks[4].

One recent trend is the move towards "worm-hole" routing in distributed systems. The basic idea here is to allow virtual circuits to be established between remote processors, and without the necessity of interrupting any intermediate nodes. While there may be a small overhead for circuit creation, subsequently all data traverses the circuit without overheads such as multiple startup costs at intermediate nodes. Once a circuit is established, communication proceeds essentially in bit-serial fashion. Frequently it suffices to create *logical* rather than physical connections. These allow messages to proceed on virtual worm-hole channels, but with the possibility that physically the channels are multiplexed. This is particularly convenient as a means for preventing dead-locks and blocking of small messages by large ones. The resulting communication performance tends to be essentially independent of distance. Worm-hole routing is utilized in the CM-2, the iPSC2, the iWarp and the Symult among others. In the case of Symult, the designers were so confident of the advantages of worm-hole routing that they abandoned a hypercube architecture from their first generation in favor of a simple two-dimensional rectangular grid. The Intel iPSC2 hypercube will give way in the near future to a rectangular-grid based iPSC3, similar in spirit to the Symult.

1.5. Software

Software for currently available parallel computers is extremely limited. In all cases manufacturers provide Fortran and C compilers, which are frequently just a single-node processor compiler. These compilers usually have no concept of parallelism or of communication capability. Typical examples are the systems supplied by Intel, Symult and NCUBE. In these systems, all communication and process control is initiated explicitly by the user, resulting in substantial code modification as well as a loss of portability of software. Typically libraries of low-level communication primitives are supplied with these systems to allow the user to initiate communications operations. The resulting software is best described as "programming in communication assembly language".

A few manufacturers have gone beyond this step by providing language extensions that capture aspects of the parallel hardware. Thinking Machines provides a parallel Fortran for their Connection Machine CM-2 computer. The compiler supports the Fortran 90 array extensions to Fortran 77, and the convention is that objects declared as arrays are understood to be distributed across the parallel processors. Communication among processors is supported by the F90 shift operations, as well as the various reduction operators such as vector sum. While the Connection Machine programming environment is remarkably elegant and user-friendly, one should point out that the task is much simplified by the SIMD nature of the hardware onto which array operations map extremely well.

Myrias Corporation (SPS-2) and Evans and Sutherland (ES-1, no longer in production) both support a virtual address space across processors. If a processor attempts to access a memory location not in its physical memory, then a page fault occurs and the appropriate memory page is fetched from the processor which has it. Myrias in particular have implemented a sophisticated mechanism for load balancing and rapid access to memory. The system attempts to localize page table information and to provide access to it in a distributed fashion. The Myrias system is the first to provide virtual shared memory on a distributed memory architecture. The ES-1 is actually a cross-bar based shared memory computer, and here again virtual memory was provided to make the memory system more transparent.

SUPRENUM supports extensions to Fortran for task control, and to assist in communication operations. In addition SUPRENUM is unique in providing a sophisticated high-level interface to the communication system. The library supports a range of grid-oriented operations that largely shield a numerical user from dealing with the communication system directly. In addition to providing powerful programming tools, such systems deliver the possibility of substantial program portability across architectures that support the common set of primitives.

One should also note the tendency to support virtual processes. This is an important aid to software development as it allows an application to simulate a larger number of processors than are physically present. Virtual processing actually can increase system throughput by allowing nodes to remain computationally active while a process is suspended waiting for memory or messages. Virtual processing is supported by the majority of systems in one form or another. Examples include iPSC, SUPRENUM, Symult, Myrias and CM-2.

2. SOME REPRESENTATIVE PARALLEL SYSTEMS

In this section we will look in more depth at the characteristics of a number of these machines. The machines currently under development have processor numbers ranging between 2 and 65,536. The machines discussed vary greatly in local processing power, ranging from a few megaflops up to 28 Gflops.

Intel iPSC

In fall 1989 Intel announced an i860-based version of the older iPSC/2 hypercube system. Two 128 processor prototypes have been shipped to Oak Ridge National Laboratory and to NASA-Ames. These iPSC/860 systems are basically standard iPSC/2 hypercubes with the node processors replaced by Intel i860 processors. In terms of raw floating point performance the peak rate is thereby increased to 60 Mflops. In practice it is unlikely that more than 40 Mflops can be realized due to the memory model used by the i860. Some simple vector type kernels, hand-coded in assembler, are currently running at from 28 to 38 Mflops/node. Well-designed Fortran programs are currently yielding about 3-4 Mflops/node due to the poor state of the i860 Fortran compilers. Several major i860 compiler efforts are underway and will undoubtedly improve substantially on the early results. Because the communication facilities of the iPSC/860 are those of the iPSC/2, the system is constrained to a maximum of 128 nodes.

While the iPSC/860 utilizes the slow iPSC/2 communication hardware and software, communication proves to be much faster on the i860 system than on the iPSC/2. This is because most of the message startup communication overhead is software overhead involved in negotiating the communication protocol. Because the i860 is so much faster than the 80386, the software overhead is correspondingly decreased. The effect is to reduce messaging time by about a factor of three.

The iPSC/860 actually supports heterogeneous boards - a mixture of i860 and 80386 boards is allowed. This permits special 80386 nodes to take advantage of the flexible interfaces to graphics, disk and other peripherals available to that processor. For example 780 Mbyte disks may be attached to such nodes via 4Mbyte/sec SCSI interface. Frame buffers, VMEbus devices and Ethernet also plug into these boards.

Intel has also announced plans to develop a rectangular grid version of the iPSC/860 - the iPSC3. This system is very similar in architecture to the Symult 2010 system. There are 8 communication paths per node, allowing 4 bidirectional channels as required for a two-dimensional grid. With the new communication structure, the iPSC will be freed from the constraint of a maximum of 128 nodes. Indeed Intel has announced a 2048 processor version of the iPSC3, called Touchstone. With a peak processing rate of 80 Gflops this will be a formidable system in the 1991 timeframe.

CMU iWarp

The iWarp computer[6] is a follow-on to the 100 Mflop Warp system developed at Carnegie-Mellon University. The key advance in the iWarp is the development of a single chip processor combining the following functions: 20 Mflops computational power, 320 Mbyte/sec memory throughput and a communication engine with a latency of only 150 nanoseconds. The processor has been implemented as a 600,000 transistor custom VLSI chip fabricated by Intel Corporation, hence the *i* in the name iWarp. Up to 64 Mbytes of memory is accessible per processor.

One important point is that the processor accomplishes 20 Mflops without pipelining. The adder unit delivers 5 Mflops (64-bit) or 10 Mflops (32-bit), non-pipelined, as does the multiplier unit. In addition the integer/logical unit delivers 20 Mips. All three units may perform simultaneously.

The system has been designed for flexibility from the start, and can be used efficiently to represent either a general purpose distributed memory computer, or special purpose systolic arrays. The initial iWarp is an 8×8 array of processors delivering 1.2 Gflops, and expected to be available in 1990, but there are plans to extend this up to 1,024 processors.

The communication facilities of iWarp are based on four input and output ports, each running at 40 Mbytes/sec. An input port of one iWarp processor may be connected directly to the output port of another processor to form a point-to-point communication network. A natural arrangement is thus to create one and two dimensional grids of processors. Because the communication processor performs independently of the numeric processor, worm-hole routing can be supported. Logical channels are supported by multiplexing of the physical communication lines, allowing for deadlock to be broken, and for long messages to be interrupted in worm-hole routing.

NCUBE2

NCUBE introduced the NCUBE hypercube at around the same time that Intel developed the iPSC1. The NCUBE is characterized by having a custom designed scalar processor, and allowing up to 1024 processors as compared to 128 for the iPSC1. Recently NCUBE has introduced the second generation NCUBE2. This system is extensible up to a 8192 node hypercube with a peak rating of 60 Gips, 27 Gflops (32-bit) and 19 Gflops (64-bit). The full size system costs over $20M, and supports 512 Gbytes of memory and 16 Tbytes of disk. 1024 processor systems have been shipped (Jan 1990) to Shell Oil and to Sandia National Laboratory. The high cost per Mflops of the NCUBE2 compared to the Intel iPSC/860 may prove to be an obstacle to marketing success.

The key advance in the NCUBE2 is the development of a 64-bit 20MHz custom processor providing high scalar performance. The communication processor supports 28 DMA channels for communication to neighboring processors, allowing 14 bidirectional connections to nearest neighbors in a hypercube.

GF-11

The GF-11 is an IBM parallel computer, designed to perform very specific scientific computations at Gflop rates. The GF-11 has 576 processors (including 64 backup processors), coupled through a three stage Benes network which can be reconfigured at every cycle in 1024 different ways by an IBM 3084 control processor. Peak processing power of 11 Gflops will allow previously uncharted computational regimes to be explored. The machine has been designed primarily for solving quantum field theory problems and is not a general purpose computer; in particular, very little system software is available. It is an SIMD architecture but with some flexibility in that the settings of local registers may be used to control the behavior of individual processors.

While difficult to program, the GF-11 has been adapted to a number of problems. As an example, a 2D fluid mechanics code is running on the system in February 1990 at 3.7 Gflops, indicating very high computational efficiency.

Connection Machine

The Connection Machine CM-1 designed by Thinking Machines, Inc., has 65,536 1-bit processors. Processors are arranged with 16 on a chip along with a communications router and associated local memory. While designed primarily for artificial intelligence work, this machine has proved to have even greater potential applications to scientific computing applications[7,8]. The more recent CM-2 computer adds 2,048 Weitek floating point processors and 2 Gbytes of memory, to provide a powerful computer for numerical as well as symbolic computing. Peak computational rate at 7MHz (the current clock rate) is theoretically about 24 Gflops (32-bit).

The CM computers are SIMD machines. Logic is supported by allowing individual processors to skip the execution of any instruction, based on the setting of a flag in their local memory. The CM communications facilities are based on a hypercube network, with a total communication bandwidth of order 3 Gbytes/sec. Communication is by worm-hole type routing. The system supports I/O to disks at up to 320 Mbyte/sec, and to graphics frame buffers at 40 Mbyte/sec.

Recently (December 1989) Thinking Machines has announced that it is developing a one million processor system with a goal of producing a TerraOps (trillion operations per second) machine by 1995.

Connection Machine software consists of parallel versions of Fortran, C and Lisp. In each case it is possible to declare parallel variables, which are automatically allocated on the hypercube. Programs execute on a front end machine, but when instructions are encountered involving parallel variables, they are executed as parallel instructions on the hypercube. The system supports the concept of *virtual processors*. A user can specify that he would like to compute with a million (or more) virtual processors, and such processors are then similar to physical processors in all respects except speed and memory size. A typical use is to assign one virtual processor per grid point in a discretization application. This provides a very convenient programming model. Parallel global memory reference is supported using both regular multi-dimensional grid notations (NEWS communication) and random access (hypercube) modes.

The Connection Machine is one of the few examples where a program from a serial machine (e.g. work-station) or from a CRAY may be moved to a parallel machine and run essentially without change. The CM-2 Fortran is Fortran-77 with the addition of the array extensions of Fortran 90. All array data types and operators are implemented as parallel objects or operators on the CM-2. In the case of Fortran-77 programs, a preliminary vectorizer is available that produces Fortran 90 as output. Because of the SIMD architecture no synchronization instructions are required.

It is extremely rare to approach the 24 Gflops peak rate of the CM-2. In practice one attains about 10% to 20% of that rate. In part this is because in addition to normal hypercube communication (e.g. to nearest neighbors in a grid) there is also extra communication required for every floating point operation. Since the floating point processors are off-chip, each of the 32 bit-serial processors that share a Weitek, must send its data to the Weitek for processing. Furthermore since the data arrives bit-serially, it needs to be "transposed" so as to be presented as floating point data to the Weitek. These two steps between the processors and the Weitek units account for much of the performance loss. Only in situations where numbers can be deposited in the 64 Weitek registers (shared by 32 processors), and then computed on for a substantial time *without leaving the Weitek*, can the theoretical speed be approached. For example, parallel polynomial evaluation proceeds at up to 20 Gflops - which ensures that transcendental functions are extremely fast on the system. Standard numerical algorithms such as relaxations or conjugate gradient iterations perform at from 2 to 4 Gflops, which is also typical of performance on regular-grid evolution problems. Despite the low efficiency, these performance numbers exceed those of a CRAY Y-MP/8, and with a price tag around $6M, the 65,536 processor system is cost-effective for classes of problems that exhibit massive SIMD data parallelism.

The CM-2 computer has established a real commercial success in the following sense. Most highly parallel machines sold to date have been purchased for experimental research at universities or major research laboratories, not for production supercomputing use. A substantial number of CM-2 machines have however been sold for production uses, including database management (Dow Jones), seismic processing (Mobil), and aircraft design (United Technologies and Lockheed) to mention a few examples. In comparing the CM-2 computer to other systems one should keep in mind that the CM-2 is (in 1990) a three to four year old system, and one should expect very sizable performance improvements in the near future.

AMT DAP

The DAP was the first massively parallel single-bit computer, and has been widely used for a range of scientific applications. Its current incarnation as the AMT 510 attached processor, provides the capability to attach a 1024 processor DAP array to any VAX or SUN computer. The 510 is a 32×32 array of processors, arranged as a two-dimensional grid and is implemented in VLSI on 16 chips. Additional busses connect all processors on each row and column and are used for broadcasts and other non-local operations. Up to 1 Mbit of memory may be installed per processor, for a combined total of 128 Mbytes. The computer is SIMD, and can execute at up to 60 Mflops, although boolean operations perform at up to 10 Gips.

MasPar MP-1

The MasPar MP-1 series (1101, 1104, .., 1116) are SIMD array machines featuring both a grid connectivity and a routing system for messages. The individual processors are 4-bit, with floating point performed in emulation mode. The systems range in size from 1K to 16K processors. The MP-1 is intermediate in design between the DAP and the CM-2. The system is priced substantially lower than a CM-2, but this is likely an advantage only for those applications not requiring floating point. Because of its intermediate position between the DAP and the CM-2, it is not clear how effective MasPar can be in capturing market share.

Myrias

The Myrias SPS-2 computer, built by Myrias Research Corp. of Edmonton, Alberta, has up to 1024 processing elements. The architecture is a hierarchical bus design, utilizing 33 Mbytes/sec busses to interconnect processors within clusters and clusters to each other. Each processor is a 32-bit Motorola 68020 microprocessor with 4 Mbytes of local memory. The architecture is a three-level hierarchical system. Processors are assembled in groups of four on a board connected among each other by a bus, along with an I/O port controller. At the second level in the hierarchy is the card cage, containing 16 processor boards and thus 64 processors, as well as one or two off-cage communication boards. Each communication board supports four off-cage links which can be connected to other cages or to the front end computer.

The SPS-2 supports a global 32-bit virtual address space. There is no concept of shared access to memory locations. Simple extensions of Fortran support parallel do and join operations. The PARDO model used by Myrias is somewhat unusual in that there are no possibilities for explicit sharing of data. A PARDO is executed by specifying a code segment to be executed and the number of child tasks to be run. Each thread of execution performs completely independently in its own address space, starting with a copy of the parents memory. Execution of a child proceeds in normal sequential mode, except that PARDO's may be nested recursively. On completion of *all* children, the memory states of the children are merged to form the new memory state of the parent. Thus a child can never affect the memory of another child, but can affect the memory state of its parent, but only after all children merge.

The rules for merging of child memories at an address on task completion are:

- If no child stored a value at the address, the location in the parent memory retains its original value.
- If exactly one child changed a value at the address, the location in the parent receives the last value from the child.
- If more than one child stores a value at the address the result is unpredicatable unless all values stored are the same.

Efficiency is maintained throughout the process by using a copy-on-write approach which ensures that most of the global address space is never really replicated.

SUPRENUM

The German SUPRENUM-1 project involves coupling up to 16 processor clusters with a network of 200 Mbit/sec. busses. The busses are arranged as a rectangular grid with 4 horizontal and 4 vertical busses. Each cluster consists of 16 processors connected by a fast bus, along with I/O devices for communication to the global bus grid and to disk and host computers. There is a dedicated disk for each cluster. Individual processors can deliver up to 20 Mflops (64-bit) of computing power (in chained operations, or 10 Mflops unchained) and support 8 Mbytes of memory. The high speed of the bus network makes this an interesting machine for a wide range of applications, including those requiring long-range communication. No more than three communication steps are ever required between remote nodes.

SUPRENUM is characterized by the best support for scientific applications to be found among the various distributed memory MIMD vendors. The effort invested in development of libraries of high-level grid and communication primitives will greatly ease the effort of moving applications to the computer, and also provides substantial high-level portability to other systems, since the communication library can be implemented in terms of low level primitives on any distributed system.

The first 64-processor system was delivered in December 1989 and will be upgraded to 256 processors during 1990. With a 5 Gflops peak rating and high realizable efficiency in some applications, it is a serious contender in the race for worlds fastest supercomputer at the current time. We have recently benchmarked the Shallow Water Equations atmospheric benchmark on the SUPRENUM-1, obtaining 4 Mflops per node[9], which equals or slightly exceeds typical iPSC2/860 performance.

3. FUTURE SUPERCOMPUTING ENVIRONMENTS: HETEROGENEOUS SYSTEMS

Over the last 20 years we have seen a gradual evolution from scalar sequential hardware to vector processing and more recently to parallel processing. No clear consensus has emerged on an ideal architecture. The trend to vector and parallel processing has been driven by the computational needs of certain problems, but the resulting systems are then inappropriate for other classes of problems. It is unlikely that in the near term this situation will be resolved, and indeed one can anticipate further generations of even more specialized processor systems appearing.

There is a simple way of avoiding the problem described above. The key is to develop seamless integrated heterogeneous computing environments. Such an environment will present to the user all of the resources he might need to avail of for any application: fast scalar, vector and parallel processors, graphics supercomputers, disk farms and interfaces to networks. All of these units would be interconnected by a high bandwidth low latency switch, which would provide transparent access between the systems. System software would present a uniform global view of the integrated resource, provide a global name space or a shared memory, and control load balancing and resource allocation.

The hardware technology is now at hand to allow such systems to be built. Two key ingredients are the recent development of fast switching systems and the development of high-speed connection protocols and hardware implementing these protocols, standardized across a wide range of vendors. We illustrate with two examples.

Recently CMU researchers have designed and built a 100 Mbit/sec switch called NECTAR which supports point to point connections between 32 processors. A Gbit/sec version of NECTAR is in the design stage. Simultaneously, various supercomputer and graphics workstation vendors have begun to develop high speed (800 Mbit/sec) interfaces for their systems. Combining these approaches we see that at the hardware level it is already possible to begin assembling powerful heterogeneous systems. As usual the really tough problems will be at the software level.

Several groups are working on aspects of the software problem. In our own group at the Center for Parallel Processing in Boulder, Colorado, we have recently developed a simple heterogeneous environment consisting of a Connection Machine CM-2 and a Stardent Titan graphics super work-station[10]. The Titan is connected with the CM-2 through the CM-2 back-end, rather than through the much slower front-end interface which is usually used for such connectivity. The object-oriented high-level Stardent AVS visualization system is then made directly available to the CM-2 user, allowing him to access and manipulate graphical objects computed on the CM-2 in real time, while the CM-2 is freed to pursue the next phase of its computation. Essentially this means that to the user, AVS is available *on* the CM-2. Porting AVS directly to the CM-2 would have been a formidable and pointless task. Furthermore the CM-2 is freed to perform the computations that it is best suited for, rather than wasting time performing hidden surface algorithms or polygon rendering.

Looking to the future, we believe that most of the research issues of heterogeneous computing will have been solved by the late 1990's, and in that time frame we would expect to see evolving heterogeneous systems coming into widespread use wherever a variety of distinct computational resources are present. In the meantime one can expect to see more limited experiments in heterogeneous environments at major research computation laboratories, such as Los Alamos National Laboratory and the NSF Supercomputer Centers in the USA. and Julich or the GMD in Germany.

References

1. M. J. Flynn, "Very high-speed computing," *Proc. IEEE*, vol. 54, pp. 1901-1909, 1966.
2. J. Schwartz, "A Taxonomic Table of Parallel Computers, Based on 55 Designs," Ultracomputer Note #69, Courant Institute, New York, 1983.
3. W.K. Giloi and S. Montenegro, "Super Interconnection Networks for Super Computers," GMD Technical Report, Berlin, 1988.
4. V. Faber and J. Moore, "High-degree Low-diameter Interconnection Networks with Vertex Symmetry: The Directed Case," Los Alamos Technical Report LA-UR-88-1051, March 1988.
5. O. McBryan and E. Van de Velde, *Hypercube Algorithms and Implementations*, SIAM J. Sci. Stat. Comput., 8, pp. 227-287, 1987.
6. Borkar, S., Cohn, R., Cox, G., Gleason, S., Gross, T., Kung, H. T., Lam, M., Moore, B., Peterson, C., Pieper, J., Rankin, L., Tseng, P. S., Sutton, J., Urbanski, J., and Webb, J., "iWarp: An Integrated Solution to High-Speed Parallel Computing," in *Proceedings of Supercomputing '88*, pp. 330-339, IEEE Computer Society and ACM SIGARCH, Orlando, Florida, Nov 1988.

7. O. McBryan, ‘‘The Connection Machine: PDE Solution on 65536 Processors,’’ *Parallel Computing*, vol. 9, pp. 1-24, North-Holland, 1988.

8. O. McBryan, ‘‘Solving PDE at 3.8 Gigaflops,’’ University of Colorado CS Dept Preprint, Sept 1987.

9. O. McBryan, *Implementation of the Shallow Water Benchmark on the SUPRENUM-1 Parallel Supercomputer,* CS Dept Technical Report, University of Colorado, Boulder, 1990.

10. L. Compagnoni, S. Crivelli, S. Goldhaber, R. Loft, O. McBryan, A. Repenning, and R. Speer, *A Simple Heterogeneous Computing Environment: Interfacing Graphics Workstations to a Connection Machine.,* CS Dept Technical Report, University of Colorado, Boulder, 1990.